A Detailed Router for Field-Programmable Gate Arrays

Stephen Brown, Jonathan Rose, and Zvonko Vranesic
Dept. of Electrical Engineering, University of Toronto, Ontario, Canada M5S 1A4

Abstract

The detailed routing of Field-Programmable Gate Arrays (FPGAs) is a new and difficult problem because the wiring segments available for routing can only be connected together in a limited number of ways. This paper presents the Coarse Graph Expansion (CGE) detailed routing algorithm for FPGAs. The algorithm has the ability to resolve routing conflicts by considering the side-effects of one connection on another, and can be used over a wide range of FPGA interconnection architectures.

CGE has been used to obtain excellent routing results for several industrial circuits with various FPGA routing architectures. The results show that CGE is able to route relatively large FPGAs in the absolute minimum number of tracks as determined by global routing, and that CGE has a linear run-time over circuit size.

1. Introduction

Field-Programmable Gate Arrays are an exciting new approach to Application Specific Integrated Circuits that reduces IC manufacturing time from months to minutes, and manufacturing costs from thousands of dollars to under $100. An FPGA has an array of logic cells connected by a general routing structure, like a Mask Programmable Gate Array, but it is programmed by the user like a PLD. The FPGA was first introduced in [Cart86], with newer versions presented in [ElGa89] and [Wong89]. The complexity of FPGAs has increased to the point where automatic design tools are essential.

A key problem in the detailed routing of FPGAs is that the successful routing of some connection may rely on the assignment of a specific wiring segment in the FPGA for that connection. If this *essential* segment is assigned to some other connection, then routing failure is guaranteed. Consider Figure 1, which shows three views of the same section of an FPGA. Each view gives the routing options for one of connections A, B, and C. In the figure, a routing switch is shown as an X, a wiring segment as a dotted line, and a possible route as a solid line. Now, assume that a router first completes connection A. If the wiring segment numbered 3 is chosen for A, then one of connections B and C cannot be routed because they both rely on the same single remaining option, namely the wiring segment numbered 1. The correct solution is for the router to chose the wiring segment numbered 2 for connection A, in which case both B and C are also routable. Although this is a simplified example, it illustrates the essence of conflicts because of limited routing options in FPGAs.

Common approaches used for detailed routing in other types of devices are not suitable for FPGAs. Maze routers [Lee61] are ineffective because they are inherently sequential and so, when routing one connection, they cannot consider the side-effects on other connections. Channel routers [Has71] are not appropriate because the general routing problem cannot be subdivided into independent channels. Note that a channel routing algorithm is used for FPGAs in [Gree90], but this only applies for the particular case of an Actel-like FPGA [ElGa89], which is arranged as rows of logic cells separated by channels.

This work was supported by NSERC Operating Grants #URF0043298, #OGP0005280 and research grants from Bell-Northern Research and ITRC.

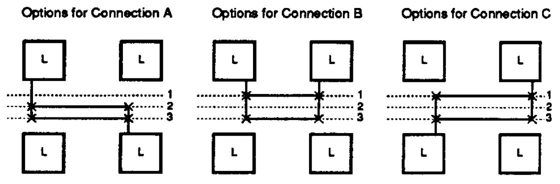

Figure 1 - *Routing Conflicts*

This paper is organized as follows: Section 2 presents the model of the FPGA, Section 3 defines the detailed routing problem, Section 4 describes the CGE algorithm, and Section 5 presents results from tests of the router.

2. The FPGA Model

The FPGA is modeled as a two-dimensional array of logic cells interconnected by vertical and horizontal routing channels, similar to [Cart86]. The FPGA comprises three major parts: the Logic (L), Connection (C), and Switch (S) blocks, as shown in Figure 2. In the figure, each L block has two pins, and there are three tracks in each routing channel. The figure also defines several terms that are used throughout the paper - note that a two-dimensional grid is overlayed on the FPGA. The L blocks are programmable cells which house the combinational and sequential logic that form the functionality of a circuit. An L block has a number of pins that may each connect to the four adjacent C blocks. Note that I/O blocks appear as L blocks on the periphery of the chip.

The C blocks are rectangular switch boxes with connection points on all four sides, and are used to connect the L block pins to the channels via programmable switches. Depending on the topology, each L block pin may be switchable to all or any fraction of the wiring segments that pass through the C block. The fewer wiring segments connectable in the C blocks, the harder the FPGA is to route. Connections may also pass straight through a C block, but in a typical routing architecture no switch would be involved for such connections.

The S blocks are also rectangular switch boxes. They are used to connect wiring segments in one channel segment to those in another. Depending on the topology of the S block, each wiring segment on one side of an S block may be switchable to all or any fraction of the wiring segments on each other side of the S block. The fewer wiring segments that can be switched to, the harder the FPGA is to route. A connection that passes through an S block may do so through a switch or it may be hardwired.

CH2924-9/90/0000/0382$01.00 © 1990 IEEE

Authorized licensed use limited to: The University of Toronto. Downloaded on July 20,2021 at 00:58:04 UTC from IEEE Xplore. Restrictions apply.

3. General Approach and Problem Definition

As in other design styles, FPGA routing is a combinatorially complex problem, requiring the usual two-stage approach of global routing followed by detailed routing. The global router used here is an adaptation of the LocusRoute global routing algorithm for standard cells [Rose90b]. The global router divides multi-point nets into two-point connections and routes them in minimum distance paths. It distributes the connections among the channels so that the channel densities are balanced.

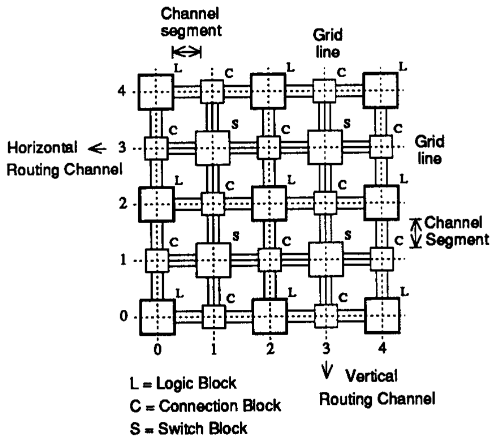

Figure 2 - *The FPGA Model*

The global router defines a coarse route for each connection, by assigning it a sequence of channel segments. Figure 3a shows a representation of a typical global route for one connection. It gives a sequence of channel segments that the global router might choose to connect some pin of a logic block at grid location 2,2 to another at 4,4. The global route is called a *coarse graph*, $G(V,A)$, where the L block at 2,2 is the root of the graph and the L block at 4,4 is the leaf. The vertices, V, and edges, A, of $G(V,A)$ are identified by the grid of Figure 2. Since the global router splits all nets into two-point connections, the coarse graphs always have a fan-out of one.

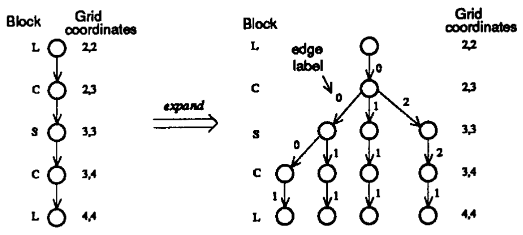

Figure 3a. Coarse graph, G Figure 3b. Expanded graph, D

Figure 3 - *A Typical Coarse Graph and its Expanded Graph*

After global routing the problem is transformed to the following: for each two point connection, the detailed router must choose specific wiring segments to implement the channel segments assigned during global routing. As this requires complete information about the FPGA routing architecture, CGE uses the details of the L, C, and S blocks, as described in the following sections.

4. The CGE Detailed Router Algorithm

CGE routes in two phases. In the first phase, it enumerates a number of alternatives for the detailed route of each coarse graph, and then in the second phase, viewing all the alternatives at once, it makes specific choices for each connection. The decisions made in phase 2 are driven by a costing function that is based on the alternatives enumerated in phase 1.

4.1 Phase 1: The Expansion of the Coarse Graphs

During phase 1, CGE *expands* each coarse graph and records a subset of the possible ways that it can be implemented. For each $G(V,A)$, the expansion phase defines a *detailed* graph, called $D(N,E)$. N are the vertices of D and E are its edges, with each edge referring to a specific wiring segment in the detailed FPGA. The edges are labelled with a number that refers to the corresponding wiring segment.

The expansion algorithm is designed to allow CGE to route arbitrary interconnection architectures by treating the procedures that define the connection topology of the C and S blocks as *black-box* functions. This approach allowed the use of CGE as a research tool in a recent paper on FPGA routing architectures [Rose90a]. The black-box function for a C block is denoted as $f_c([d_1,d_2,l],d_3)$ and for an S block as $f_s([d_1,d_2,l],d_3)$. The parameters in square brackets define an edge that connects vertex d_1 to vertex d_2, using a wiring segment labelled l. Such an edge is later referred to as e, where $e=(d_1,d_2,l)$. The parameter d_3 is the successor vertex of d_2. The task of the function call can be stated as: "If the wiring segment numbered l is used to connect vertex d_1 to d_2, what are the wiring segments that can be used to reach d_3 from d_2?" The function call returns the set of edges that answer this question. Figure 3b shows an expanded graph for the coarse graph of Figure 3a.

The graph expansion process for each coarse graph proceeds as follows:

```
Create D and give it the same root as G. Make the immediate
    successor to the root of D the same as for the root of G.
While traversing D breadth first, expand each added vertex
    according to:
    Expand a C vertex in D by calling $f_c(e_C,n)=Z$. $e_C$ is the
        edge in D that has already been chosen to connect to
        C from its predecessor. n is the required successor
        vertex of C (in G) and Z is the set of edges returned
        by $f_c()$. The call to $f_c()$ adds $|Z|$ edges to D.
    Expand an S vertex in D by calling $f_s(e_S,n)=Z$. $e_S$ is the
        edge in D that has already been chosen to connect to
        S from its predecessor. n is the required successor
        vertex of S (in G) and Z is the set of edges returned
        by $f_s()$. The call to $f_s()$ adds $|Z|$ edges to D.
```

Although the above description implies that all possible paths in an FPGA are recorded during the expansion process, this is not practical because the number of paths can be very large in some architectures. Consequently, CGE reduces the number of paths by *pruning* as it expands. At regular intervals, as the coarse graph is expanded, heuristics are used to discard selected paths. The heuristics choose which paths to keep based on the wiring segments they use. To help prevent routing conflicts, when choosing between two wiring segments the heuristics will keep the one that has been used in fewer expanded graphs thus far. Note that when paths are discarded because of pruning, they are not necessarily abandoned permanently by the router. In phase 2, as CGE chooses connections,

Authorized licensed use limited to: The University of Toronto. Downloaded on July 20,2021 at 00:58:04 UTC from IEEE Xplore. Restrictions apply.

if routing conflicts consume all of the alternatives for some graph, CGE re-invokes the graph expansion process to obtain a new set of paths if some exist.

4.2 Phase 2: Connection Formation

After expansion, each $D(N,E)$ may contain a number of alternative paths. CGE places all the paths from all the expanded graphs into a single path list. Based on a costing function, CGE then selects paths from the list; each selected path defines the detailed route of its corresponding connection. Because the costing function allows it to consider all the paths at once, CGE is said to route the connections 'in parallel'. Phase 2 proceeds as follows:

```
Put all the paths in the expanded graphs into the path-list
While the path-list is not empty
    If there are paths in the path-list that are known to be
        essential
        Select the essential path that has the lowest cost.
    Else
        Select the path with the lowest cost
    Mark the graph corresponding to the selected path as
        routed - remove all paths in this graph from the
        path-list.
    Find all paths that would conflict with the selected path
        and remove them from the path list (see Note). If a
        connection loses all of its alternative paths, re-
        expand its coarse graph - if this results in no new
        paths, the connection is deemed unroutable.
    Update the cost of all affected paths.
Endwhile
```

Note: When a wiring segment is chosen for a particular connection, it and any other wiring segments in the FPGA that are hardwired to it must be eliminated as possible choices for other nets. This requires a function analogous to $f_c()$ and $f_s()$ that understands the connectivity of a particular FPGA configuration. CGE calls this routine *update*(e) - the parameter e is an edge in the selected path. *Update*$()$ returns the set of edges that are hardwired to e.

4.2.1 Cost Function Design

Each edge in the expanded graphs has a cost, $c(e)$, which accounts for the competition between different nets for the same wiring segments. The cost of a path is simply the sum of the costs of its edges.

CGE's cost serves two purposes:

1. It is used to identify a path that is *essential* for a connection. Such a connection has only one path remaining in the detailed FPGA, because previous path selections have consumed its alternatives.
2. It allows CGE to select a path such that it has the least negative effect on the remaining connections, in terms of routability. The cost deters the selection of paths that contain wiring segments that are in great demand.

To derive a cost expression, consider an edge e_1 in G_1 that also appears in G_2 (which is in a different net), where it is called e_2. The more desirable e_2 is in G_2, the higher should be the cost of e_1. This desirability depends on the number of alternatives that exist in G_2 for e_2. To reflect the desirability, the term $alt(e)$ is defined as the number of edges that are in parallel with e in its graph - i.e. that could be used instead of e. Now, in general, consider an edge e that has j other occurrences in the forest of expanded graphs. The cost of e is defined as

$$c(e) = \sum_j 1 / alt(e_j),$$

Because of the summing process in $c(e)$, the more graphs e occurs in, the higher will be its cost. This reflects the fact that e is an edge that is in high demand and urges CGE to avoid using e when there are other choices. Note that an edge that only appears in its own graph will have a cost of 0. For the special case when $alt(e_j)$ is 0, e_j is an edge that is essential to the associated connection because there are no alternatives. In this case, any path in the graph that uses e_j is called *essential*. When the calculation of a cost reveals that a path is essential, CGE gives it the highest priority for routing.

5. Results

CGE has been used to route several industrial circuits. The routing results shown in this section are based on five circuits from four sources: Bell-Northern Research, Zymos, and two different designers at the University of Toronto. Table 1 gives the name, size (number of logic blocks, and two-point connections), source and the function of each circuit. For these results, the L block used is the result of a previous study [Rose89], and the S and C blocks will be described in the next sub-section. Results are presented for a routing architecture similar to a commercial FPGA.

Circuit	#Blocks	#Conn	Source	Type
BUSC	109	392	UTD1	Bus Cntl
DMA	224	771	UTD2	DMA Cntl
BNRE	362	1257	BNR	Logic/Data
DFSM	401	1422	UTD1	State Mach.
Z03	586	2135	Zymos	8-bit Mult

Table 1 - *Experimental Circuits*

5.1 FPGA Routing Structures

Since the routability of an FPGA is determined by the topology and flexibility of its S and C blocks, those used in the tests of the algorithm are presented here. The general nature of the S block is illustrated in Figure 4a. Its flexibility is defined by a parameter called F_s, which defines the total number of connections offered to each wiring segment that enters the S block. For the example shown in Figure 4a, the wiring segment at the top left of the S block can connect to three other wiring segments, and so F_s is 3. Although not shown, the other wiring segments are similarly connected.

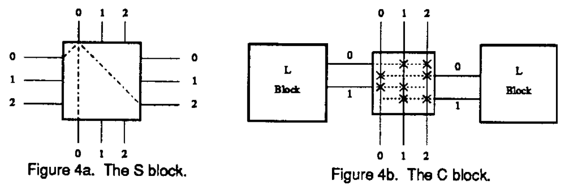

Figure 4 - *Definitions of S and C Block Flexibility*

Figure 4b illustrates the test C block. The tracks pass uninterrupted through it and are connected to L block pins via a set of switches. The flexibility of the C block, F_c, is defined as

Authorized licensed use limited to: The University of Toronto. Downloaded on July 20,2021 at 00:58:04 UTC from IEEE Xplore. Restrictions apply.

the number of tracks that each L block pin can connect to. For the example shown in the figure, each L block pin can connect to 2 vertical tracks, and so F_c is 2.

5.2 Routing Results

The familiar yardstick of channel density is used as a measure of the quality of the detailed router. Table 2 gives the largest channel density of all the channels for each circuit, as determined by the global routing. However, the global router assumes complete flexibility in the FPGA routing structures ($F_s = 3W$ and $F_c = W$, where there are W tracks per channel), and this is not practical for real FPGAs. For the results in Table 2 the FPGA parameters are based on the Xilinx 3000 series [Xil89] FPGAs ($F_s = 6$, $F_c = 0.6W$). The table gives the minimum number of tracks needed for CGE to route 100 percent of the connections. The values for W are slightly greater than the global router minimum, which are excellent results considering the low flexibility of the FPGA. Note that if F_c is increased to 0.8W, that CGE achieves the absolute minimum number of tracks for all the circuits.

Circuit	F_s	Channel density	W for 100% routing	W for 'maze'	CPU secs.
BUSC	6	9	10	15	25
DMA	6	10	10	15	59
BNRE	6	11	12	20	122
DFSM	6	10	10	18	103
Z03	6	11	13	18	215

Table 2 - *CGE Minimum W for 100 % routing ($F_c = 0.6W$)*

For comparison purposes, the same problems have also been routed using CGE with its cost facility disabled. In this mode CGE is basically a sequential router, much like a maze router. The connections are ordered for the 'maze' router by descending length because the longest connections require the most routing resources and intuitively should be the hardest to route. The second from the right column in Table 2 gives the number of tracks that the 'maze' router needed to achieve 100 percent routing. These results show that the 'maze' router requires an average of 68 percent more tracks than CGE. This shows that resolving routing conflicts is important and that CGE addresses this issue well. Figure 5 shows the detailed routing for circuit BUSC, with the FPGA parameters in Table 2; the L blocks are shown as solid boxes, whereas the S and C blocks are dashed boxes.

5.3 Memory Requirements and Speed of CGE

For the examples used here CGE needs between 1.5 and 7.5 Mbytes of memory. As shown in the rightmost column of Table 2, experimental measurements show that CGE is a linear-time algorithm, requiring from 25 to 215 SUN 3/60 CPU seconds for the smallest to the largest of the example circuits.

6. Conclusions and Future Work

This paper has described the implementation of a new kind of detailed routing algorithm that can be used to route a wide range of FPGA routing architectures. The algorithm is able to consider the side-effects that routing decisions made for one connection may have on another, and thus resolve routing conflicts. In future research, routing delay optimization will be added to CGE.

Acknowledgments

The authors would like to thank David Lewis and Paul Chow for suggestions on the design of the router.

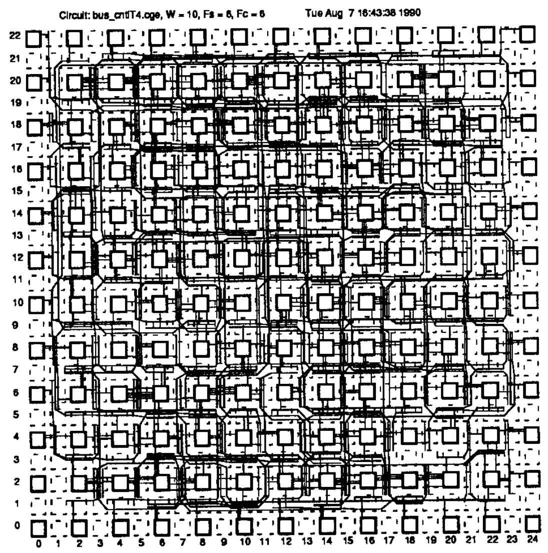

Figure 5 - *The Detailed Routing of Circuit BUSC*

6. References

[Cart86] W. Carter et. al, "A User Programmable Reconfigurable Gate Array," *Proc. 1986 CICC*, May 1986, pp. 233-235.

[ElGa89] A. El Gamal, et. al, "An Architecture for Electrically Configurable Gate Arrays," *IEEE JSSC* Vol. 24, No. 2, April 1989, pp. 394-398.

[Gree90] J. Greene, V. Roychowdhury, S. Kaptanoglu, and A. El Gamal, "Segmented Channel Routing," *Proc. 27th DAC*, pp. 567-572, June 1990.

[Has71] A. Hashimoto, and J. Stevens, "Wire routing by optimizing channel assignment within large apertures," *Proc. 8th DAC*, pp. 155-163, 1971.

[Lee61] C. Lee, "An algorithm for path connections and its applications," *IRE Trans. on Electronic Computers*, VEC-10, pp. 346-365, Sept. 1961.

[Rose89] J.S. Rose, R.J. Francis, P. Chow, and D. Lewis, "The Effect of Logic Block Complexity on Area of Programmable Gate Arrays," *Proc. 1989 CICC*, May 1989, pp. 5.3.1-5.3.5.

[Rose90a] J. Rose, and S. Brown, "The Effect of Switch Box Flexibility on Routability of Field Programmable Gate Arrays," *Proc. 1990 CICC*, pp. 27.5.1-27.5.4, May 1990.

[Rose90b] J. Rose, "Parallel Global Routing for Standard Cells," *IEEE Transactions on CAD* Vol. 9, No. 9, September 1990.

[Wong89] S.C. Wong, H.C. So, J.H. Ou, J. Costello, "A 5000-Gate CMOS EPLD with Multiple Logic and Interconnect Arrays," *Proc. 1989 CICC*, May 1989, pp. 5.8.1 - 5.8.4.

[Xil89] *The Programmable Gate Array Data Book*, Xilinx Co., 1989.

Authorized licensed use limited to: The University of Toronto. Downloaded on July 20,2021 at 00:58:04 UTC from IEEE Xplore. Restrictions apply.